Software Technology to Enable Reliable High-Performance Distributed Disk Arrays

First Year Progress Report

Michael S. Warren, Chris L. Fryer, M. Patrick Goda, Ryan Joseph

Mail Stop B227
Theoretical Astrophysics
Los Alamos National Laboratory
Los Alamos, NM 87545

Abstract

It is currently possible to construct a single-node RAID storage system with a 2 Terabyte capacity using commodity serial-ATA hard disk drives for less than $4000. Within a very few years, a cluster of such systems (a distributed disk array) will be able to provide over a petabyte of storage for less than 1 million dollars. Obtaining reliablity and good performance from such a system is the focus of our project.

Over the past year, we have established a number of testbed systems containing over 50 Terabytes of storage. Roughly half of this amount is in the 288 processor Space Simulator Beowulf cluster, with the remainder in a variety of RAID systems. Significant progress to date includes detailed failure statistics on a variety of disk drives, performance benchmarks on a number of different systems, and modifications to the Linux Network Block Device (NBD) driver to support RAID-5 arrays across multiple cluster nodes.

1 Introduction

The advent of commodity microprocessors with adequate floating-point performance and low-priced fast ethernet switches contributed to the emergence of Beowulf clusters in the mid-90s. We are currently poised for a similar advance in distributed disk arrays (DDAs), due to the dramatic decline in the price of commodity disk drives.

The cost per Gbyte for 7200 RPM IDE disk drives is currently less than $1.00. Several groups have demonstrated fault-tolerant Terabyte RAID servers for a total cost of under $2000 per Terabyte. Used in a parallel cluster environment, multi-terabyte disk arrays with achievable read/write bandwidths that greatly exceed available Gigabit local and wide-area networking technology are possible. Additionally, the greater CPU/storage ratio in a DDA offers techniques which are not possible in traditional RAID arrays.

While projects such as the parallel virtual file system have demonstrated clear utility, they lack the fault-tolerance that could be obtained via the efficient calculation and storage of parity or mirroring information between nodes (analogous to RAID techniques within a node). This additional functionality would add orders-of-magnitude to the reliability of mass storage on clustered systems.

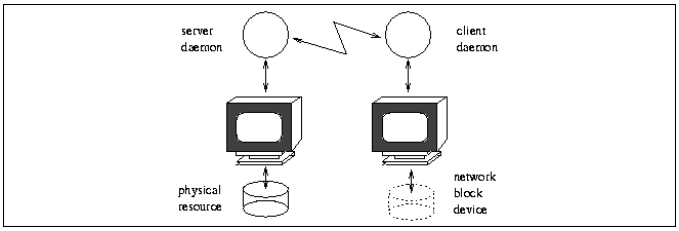
Also, while disk areal density has been improving at an annual rate of about 100% per year, disk latency has been improving 10%, so disks are becoming increasingly unbalanced in terms of capacity and latency. By intelligently replicating and caching data in a DDA, it is possible to reduce latencies to access terabytes or more of data by an order of magnitude.

2 Progress During Year 1

2.1 Network Block Device

This project depends fundamentally on scaling disk storage from a single system to a parallel system. The networking abstraction used to communicate between systems is an important facet of this approach.

Recently, the iSCSI specification was developed in order to standardize communication with network attached SCSI devices. Our summer intern, Ryan Joseph, investigated all of the major iSCSI implementations to see if they would provide a solid foundation to proceed from. The conclusions from this research were that iSCSI was over-engineered, and no robust implementations currently exist. The two most successful implementations were the UNH iSCSI and Linux iSCSI packages, but they were not compatible with each other. Additionally, use of iSCSI with ATA devices would require an additional layer of software.



However, the “Network Block Device” exists as a standard part of the Linux kernel, and provides the functionality required for network attached storage. An NBD is ”a long pair of wires”. It makes a remote disk on a different machine act as though it were a local disk on your machine. It looks like a block device on the local machine where it’s typically going to appear as /dev/nda. The remote resource doesn’t need to be a whole disk or even a partition. It can be a file. A graphic representation of an NBD is shown in Figure 2.1

The NBD system works by emulating a block device on the client side, while actual requests to that device are passed over the network to the true block device or file on one or many NBD servers. NBD consists of two user-space programs and one kernel-space module: the two user-space programs are a client and server, respectivley, while the module loaded into any kernel that wishes to be a client for NBD.

The most common sequence of operations required to start an NBD system consists of:

- Start the user-space nbd-server on all machines 0..n that wish to act as a block device server. This process simply waits for socket requests and fulfills them by accessing the device (ie: /dev/hda) or file (ie: /tmp/nbd-device).

- Load the kernel module nbd.o into a running kernel on a machine that will act as the ”head” or ”master” (machine M) for the NBD system: the kernel module registers as major device no. 43, and the MAKEDEV.enbd script will create devices in /dev with this major number. This kernel driver sets up NBD as a block device, creating the standard kernel request queue for NBD.

- Run one instance of nbd-client on machine M for each running nbd-server on machines 0..n. Each instance of nbd-client will need to be attached to a different device in /dev with major number 43, ie: each nbd-client is a different minor numbered device (the devices are usually named /dev/nbN or /dev/ndxN).

- Operate on the NBD devices as if they were actual hardware block devices: the linux RAID layer only requires that devices it operates on be standard block devices, so NBD devices make perfect RAID elements.

  When you start NBD, this is (for the most part) what happens in the code:

- nbd-client sets up a client socket and goes through a large amount of error-checking to ensure that the proper conditions are met for NBD to be used. Just after the client forks into the background, it sends an `NBD_DO_IT` ioctl to the server.

- nbd-server connects the client-end socket, and enters `mainloop() serveconnection()`. Since Linux block devices are fairly simple (responding usually only to reads and writes), `mainloop()` listens for a read, write, or disconnect request from the client kernel driver, and satifies the request by accessing the device.

- Only the client side, the `NBD_DO_IT` ioctl that was dispatched by nbd-client upon a succesful connection to an NBD server causes the kernel module to run `nbd_do_it()` (a function call this time), which enters an infinite loop in which it waits for requests from the kernel block device layer to process. If the NBD kernel module recieves a request from the kernel (ie: a

read or write request), it goes through the normal block device motions (reading from the request queue, running `b_end_io()`), but instead of accessing a local block device, NBD dispatches a request over the connected network socket (which has been set in the kernel module by the `NBD_SOCK_SET` ioctl from user-space).

- If a socket request fails (which can happen if the network is fairly congested) in the kernel module, the module assumes that it is a fatal error and closes the request queue and socket. This can be a large problem when using NBD devices in a RAID array, as the cause of the failure is usually just network congestion that will clear up. However, the RAID layer has not written to be aware of network devices, so there is a fair amount that needs to be done to keep the RAID layer from failing constantly when using NBD devices as elements.

The last item was resolved by developing modifications to the nbd driver to make it more robust. These modifications were successful enough to create a number of large NBD devices using the Space Simulator. Performance results are given in the benchmarks section below.

2.2 The Space Simulator

The Space Simulator [1, 2] is our third generation Beowulf cluster. The first was Loki [3], which was constructed in 1996 from 16 200 MHz Pentium Pro processors for $50k. Loki was among the earliest generation of Beowulf clusters [4], and was the first to be recognized with the Gordon Bell price/performance award [5]. Loki was followed by the Avalon cluster [6], which used 144 alpha processors and cost about $300k. Avalon also won a Gordon Bell prize [7] and was ranked as the 113th fastest computer in the world in June 1998 [8]. The Space Simulator follows the same basic architecture as our previous machines, but is the first to use Gigabit Ethernet as the network fabric, and requires significantly less space than a cluster using typical ATX cases. In the context of this project, the Space Simulator provides a testbed with about 20 Terabytes of storage in a standard Beowulf architecture.

The funds to purchase the Space Simulator became unexpectedly available in mid-July of 2002. Fiscal constraints required the system to be delivered by September 31. This left little time for benchmarking systems and testing components. Our goal was to purchase a computer which would obtain the highest performance possible on the astrophysics codes we wanted to run, within the budget we were alloted. It had to be delivered within one month. The machine also had to be reliable and maintainable enough that our very limited system administration resources would be capable of keeping the machine operational. We estimated the amount of cooling capacity available would limit the cluster to about 35 kW of power dissipation. The Space Simulator architecture (see Table 1) was defined by mid-August of 2002, based on the Shuttle XPC chassis.

Our limited preliminary benchmarking demonstrated our codes were faster on Intel processors than on similarly priced AMD processors and that higher performance RDRAM did not justify its extra cost over DDR SDRAM. The Green Destiny architecture [9] was ruled out on price-performance grounds, since our space, power and cooling were not sufficiently constrained. The XPC system was selected due to its small size, the elimination of the failure-prone CPU fan, and its ability to support the relatively new 533 MHz front side bus for the Intel Pentium 4 architecture. The main disadvantages of the system were that it provided only a single 32-bit 33 MHz PCI expansion slot, and 10% of its memory bandwidth was shared with the on-board video controller.

Qty.	Price	Ext.	Description
294	280	82,320	Shuttle SS51G mini system (bare)
294	254	74,676	Intel P4/2.53GHz, 533MHz FSB, 512k cache
588	118	69,384	512Mb DDR333 SDRAM (1024Mb per node)
294	95	27,930	3com 3c996B-T Gigabit Ethernet PCI card
294	83	24,402	Maxtor 4K080H4 80Gb 5400rpm Hard Disk
294	35	10,290	Assembly Labor/Extended Warranty
		4,000	Cat6 Ethernet cables
		3,300	Wire shelving/switch rack
		1,378	Power strips
1		186,175	Foundry FastIron 1500+800, 304 Gbit ports
Total		$483,855	$1646 per node 5.06 Gflop/s peak/node

Table 1: Space Simulator architecture and price (September, 2002). The total cost per node was $1646, with $728 (44%) of that figure representing the Network Interface Cards and Ethernet switches

It is interesting to note that the volume occupied by a Shuttle XPC chassis (30x20x18.5 cm or 0.011 cubic meters) is nearly the same as a 1U rackmount chassis (19x23.6x1.75 inches or 0.013 cubic meters). We have found the ability to easily remove a node from a shelf to be a major advantage over the complexities of a typical 1U rackmount on rails solution. Overall, the architecture which was chosen has proven to be an excellent solution, and the limited amount of benchmarking and testing prior to ordering the system has not resulted in problems.

The Space Simulator was selected as a finalist for the Gordon Bell price/performance award. Our results were quoted from a typical cosmology run, which took place over a continuous 24 hour period on 250 processors. The code saved 1.5 Tbytes of data, and performed 10^{16} floating point operations, for an average I/O rate of 417 Mbytes/sec and 112 Gflop/s. I/O was done in parallel to and from the local disk on each processor, with a peak I/O rate which was near 7 Gbytes/sec.

2.3 Benchmarks

We evaluated a number of systems in order to quantify the performance of single disks, as well as different types of hardware and software RAID5 systems. Our initial results indicate the best price/performance system uses Serial ATA disks with a 3ware controller and Linux software RAID (Table 2.

2.4 Hardware Monitoring

We have gathered nearly 1 year of disk failure statistics, including the internal statistics gathered by the internal `SMART` system included with all modern disk drives. An example of the types of statistics which can be monitored are listed below.

```
SMART Attributes Data Structure revision number: 16
Vendor Specific SMART Attributes with Thresholds:
```

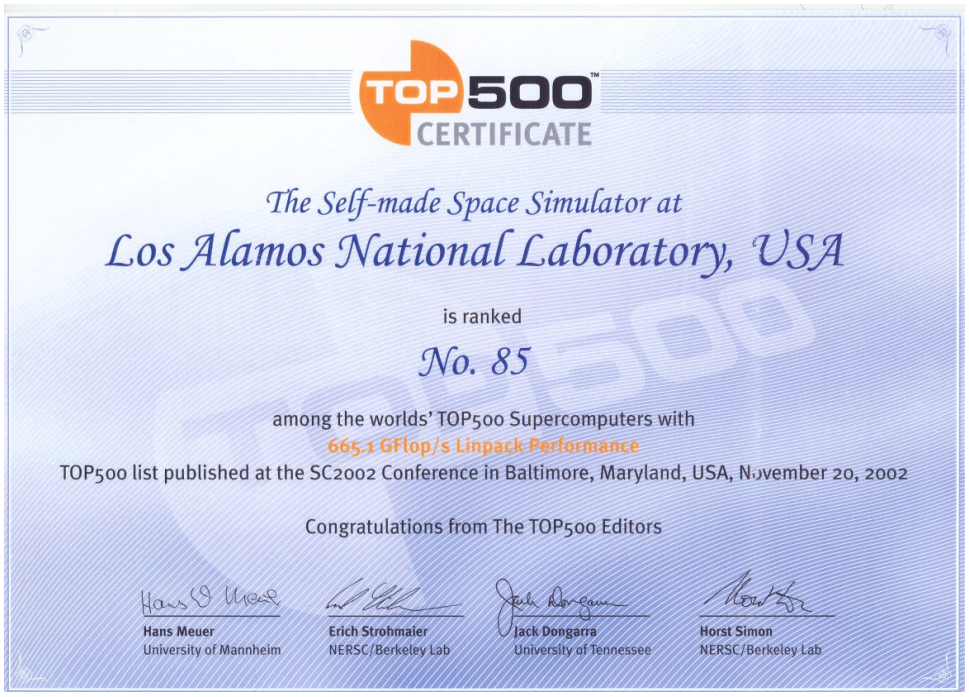


Figure 1: The Space Simulator ranked at #85 on the 20th `TOP500` list of the fastest computers in the world, as determined by the Linpack benchmark. Performance of 665.1 Gflop/s was obtained on 288 processors in October 2002. In April 2003, we obtained a higher figure of 757.1 Gflop/s through the use of a slightly faster version of ATLAS and using LAM instead of `mpich`. This ranks at #88 on the 21st `TOP500` list, and that performance would have ranked the Space Simulator at #69 on the 20th `TOP500` list. We believe our results are the first example of a machine in the `TOP500` with price/performance of better than 1 dollar per Mflop/s (we obtain 63.9 cents per Mflop/s, or $639 per Gflop/s).

	Write	Re-Write	Read	Re-Read
IDE RAID-5	80.12	115.48	210.32	204.69
Outer Track IDE RAID	93.31	123.44	269.94	276.45
SATA Hardware RAID	27.44	20.09	62.73	61.24
SATA Software RAID, ext2	95.99	81.28	230.18	226.63
SATA Software RAID, ext3	77.43	79.45	229.31	220.09
NBD baseline, 1kb BS	20.76	19.14	78.63	79.19
NBD w/ 4kb BS	-	22.38	22.33	38.70
NBD w/ 512b BS	20.24	19.70	27.84	27.23
14-disk NBD RAID-5	91.23	93.46	56.38	56.78
11-disk NBD RAID-5	44.92	54.92	75.82	67.99
5-disk NBD RAID-5	16.34	16.47	81.83	72.66
tmpfs RAID-0	107.96	125.31	95.78	96.77
tmpfs RAID-5	81.07	87.52	90.86	90.90

Table 2: Hardware Storage Benchmarks (all data in Mbytes/sec)

```
ID# ATTRIBUTE_NAME            VALUE WORST THRESH TYPE      UPDATED  RAW_VALUE
  1 Raw_Read_Error_Rate       200   200   051    Pre-fail  Always    0
  3 Spin_Up_Time              102   100   021    Pre-fail  Always    5691
  4 Start_Stop_Count          100   100   040    Old_age   Always    10
  5 Reallocated_Sector_Ct     200   200   140    Pre-fail  Always    0
  7 Seek_Error_Rate           200   200   051    Pre-fail  Always    0
  9 Power_On_Hours            095   095   000    Old_age   Always    4051
 10 Spin_Retry_Count          100   253   051    Pre-fail  Always    0
 11 Calibration_Retry_Count   100   253   051    Pre-fail  Always    0
 12 Power_Cycle_Count         100   100   000    Old_age   Always    10
196 Reallocated_Event_Count   200   200   000    Old_age   Always    0
197 Current_Pending_Sector    200   200   000    Old_age   Always    0
198 Offline_Uncorrectable     200   200   000    Old_age   Always    0
199 UDMA_CRC_Error_Count      200   253   000    Old_age   Always    0
200 Multi_Zone_Error_Rate     200   200   051    Pre-fail  Offline   0
```

The SMART system allows us to measure the temperature of each drive. A graphical representation of the entire cluster is shown in figure 2. The chilled air enters the cluster from the left side, so there is strong temperature gradient across the nodes. Since the temperature varies significantly, in a known way, it may be possible to quantify the effect of temperature on the probability of failure for a typical disk drive.

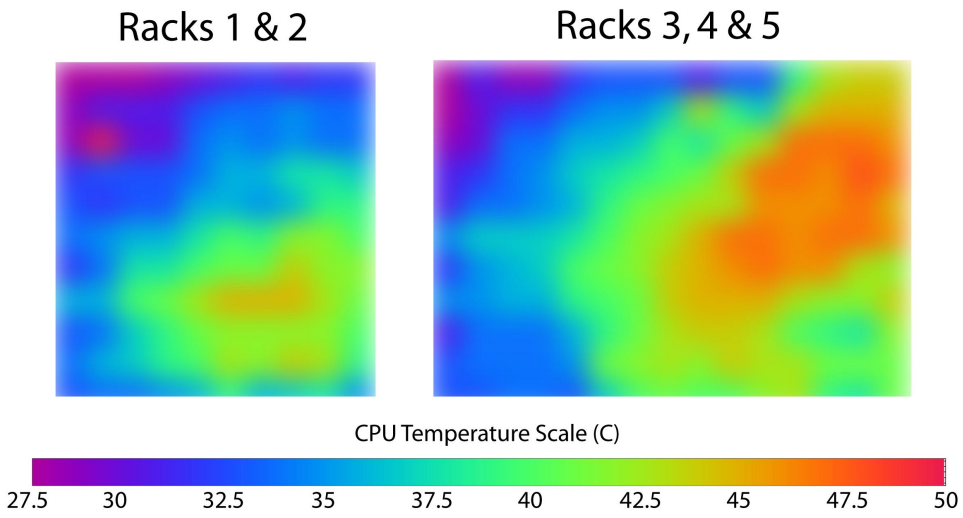


Figure 2: A smoothed representation of the temperature across the cluster.

Overall, about 10% of the drives in the Space Simulator cluster have failed in their first year of operation. This is significantly higher than the quoted MTBF for these drives. We plan to investigate the precise definition of "failure" used in the vendor's numbers.

2.5 Reference "I/O Brick" design

We investigated a number of storage systems in order to determine the optimal "building block" for larger systems. Our final reference design is listed in Table 3, which has a storage capacity

Figure 3: Firewire and DVD

Figure 4: AICIPC Hot-Swap Chassis (Parallel ATA and Serial ATA)

of 2 Tbytes at a cost of about $4000. We purchased 10 such systems, which will be the testbed hardware for further software development.

3 Plans for Next Year

Linux version 2.4 has serious limitations:

- Limit of 27 devices in a RAID
- Limit of 2 Tbytes per filesystem
- A few bad sectors on a disk cause the RAID driver to declare the entire disk “failed”
- Rebuilding arrays is time-consuming. It might be possible to optimize this by extending the journal already used for the filesystem

Figure 5: Serial ATA solutions

Qty.	Price	Ext.	Description
1	260	260	Intel P4 Processor 2.8GHz, 533MHz FSB
8	307	2456	Western Digital WD2500JD 250GB SATA 7200RPM
1	530	530	3ware 8506-8 RAID card
1	322	322	TYAN GC-SL S2707G2N-533 with Dual GigE and PCI-X
2	120	240	Corsair 2x512MB DDR266 PC2100 ECC Registered memory
1	150	150	Supermicro 5 Bay Hot-Swapable SATA HDD Enclosure
1	75	75	Mid-Tower case, 4x5.25 exposed, 4x3.5 int., extra fans
1	94	94	Enermax EG465P-VE (FCA) 431W Power Supply
1	80	80	Assembly
Total		$4207	$2.10 per Gbyte

Table 3: Mid-tower storage system pricing, August 2003

We plan to use the new Linux version 2.6 kernel to bypass many of these limitations, and our work will focus on extending the network block device to enhance its performance and scalability.

We will also work on demonstrating techniques for cheaply and efficiently duplicating and transporting multi-terabyte datasets. Currently, 1 Terabyte of data stored on IDE disk drives weighs 6 lbs. Using an overnight courier, that data can be transported across the country for $25. To transport the same amount of data over the Internet in the same amount of time requires a constant 100 Mbits/sec. Current wholesale costs for Internet bandwidths are in the range of tens of thousands of dollars per month for a committed 100 Mbit connection. Even a slower 10 Mbit connection would cost more in a month than the cost of the disk drives to store a Terabyte, and the cost of the disk hardware would be amortized over multiple transport cycles. The obvious conclusion is that it is 10x cheaper and likely faster to move Tera-scale datasets via physical media, rather than the Internet.

4 Publications and Presentations

M. S. Warren, C. L. Fryer, and M. P. Goda. The Space Simulator. In *Proceedings of CWCE '03*, San Jose, 2003.
M. S. Warren, C. L. Fryer, and M. P. Goda. The space simulator: Modeling the universe from supernovae to cosmology. In *Proceedings of the ACM/IEEE SC2003 Conference*, New York, 2003. ACM Press.
Chris L. Fryer and Michael S. Warren. The collapse of rotating massive stars in 3-dimensions. *Ap. J.*, 2003. (submitted).
Invited talk at ClusterWorld conference and Expo, San Jose CA June 2003.
Seminar at Caltech Center for Advanced Computing Research, Pasadena CA, August 2003.
Gordon Bell Prize Finalist Presentation at SC2003, Phoenix AZ, November 2003.

References

[1] M. S. Warren, C. L. Fryer, and M. P. Goda. The Space Simulator. `http://space-simulator.lanl.gov/`, 2002.

[2] M. S. Warren, C. L. Fryer, and M. P. Goda. The Space Simulator. In *Proceedings of CWCE '03*, San Jose, 2003.

[3] M. S. Warren and M. P. Goda. Loki – commodity parallel processing. `http://loki-www.lanl.gov/`, 1996.

[4] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer. BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 1995 International Conference on Parallel Processing (ICPP)*, pages 11–14, 1995.

[5] M. S. Warren, J. K. Salmon, D. J. Becker, M. P. Goda, T. Sterling, and G. S. Winckelmans. Pentium Pro inside: I. A treecode at 430 Gigaflops on ASCI Red, II. Price/performance of $50/Mflop on Loki and Hyglac. In *Supercomputing '97*, Los Alamitos, 1997. IEEE Comp. Soc.

[6] M. S. Warren, A. Hagberg, D. Moulton, and D. Neal. The avalon beowulf cluster. `http://cnls.lanl.gov/avalon`, 1998.

[7] M. S. Warren, T. C. Germann, P. S. Lomdahl, D. M. Beazley, and J. K. Salmon. Avalon: An Alpha/Linux cluster achieves 10 Gflops for $150k. In *Supercomputing '98*, Los Alamitos, 1998. IEEE Comp. Soc.

[8] J. J. Dongarra, H. W. Meuer, and Strohmaier E. TOP500 supercomputer sites. *Supercomputer*, 13(1):89–120, 1997.

[9] M. S. Warren, E. H. Weigle, and W. Feng. High-density computing: A 240-processor Beowulf in one cubic meter. In *SC '02*, Los Alamitos, 2002. IEEE Comp. Soc.

[10] M. S. Warren, C. L. Fryer, and M. P. Goda. The space simulator: Modeling the universe from supernovae to cosmology. In *Proceedings of the ACM/IEEE SC2003 Conference*, New York, 2003. ACM Press.

[11] Chris L. Fryer and Michael S. Warren. The collapse of rotating massive stars in 3-dimensions. *Ap. J.*, 2003. (submitted).

[12] J. Salmon and M. S. Warren. Parallel out-of-core methods for N-body simulation. In *8th SIAM Conf. on Parallel Processing for Scientific Computing*, Philadelphia, 1997. SIAM.